Pembelajaran Dalam (Deep Learning)

Chapter 1
Artificial Neural Networks (ANN)

Genap TA 2022/2023

FYS, ITQ | Maret 2023

Artificial Neural Networks

1. Pendahuluan Pembelajaran Mesin dan Pembelajaran Dalam (CLO 1)
2. Programming Python: Dasar Numpy, Pandas, Matpotlib, dll (CLO 3)
3. Dasar Regresi dan Klasifikasi (CLO 1)
4. **Artificial Neural Networks (ANN) (CLO 2/3)**
5. Multi-Layer Perceptron (MLP) (CLO 2/3)
6. TensorFlow (CLO 3)
7. TensorFlow (+Kinerja Machine Learning) (CLO 3/1)
8. TensorFlow Lanjutan (CLO 3)
9. Convolutional Neural Networks (CNN) (CLO 2/3)
10. CNN Lanjutan (CLO 2/3)
11. Recurrent Neural Networks (RNN) (CLO 2/3)
12. RNN Lanjutan (CLO 2/3)
13. Generative Adversarial Networks (GANs) (CLO 2/3)
14. GANs lanjutan (CLO 2/3)
15. Aplikasi Object Detections (CLO 2/3)
16. UAS (CLO 2/3)

Artificial Neural Networks

Course Learning Outcomes (CLO)

- CLO 2: Mahasiswa dapat mendeskripsikan ide dasar dan cara kerja algoritma pembelajaran dalam (deep learning)
- CLO 3: Mahasiswa dapat mengimplementasikan algoritma pembelajaran dalam yang populer menggunakan perangkat lunak

Artificial Neural Networks

Tujuan Perkuliahan

- Pada bagian ini akan dipelajari dua contoh algoritma untuk klasifikasi sebagai dasar-dasar memahami Deep Learning: **Perceptron** dan **Adaptive Linear Neurons (Adaline)**
- Mengimplementasikan algoritma menggunakan Python dengan tujuan membangun intuisi algoritma-algoritma machine learning

Artificial Neural Networks

Outline:

A. Artificial Neurons

- Neuron Biologis
- Komputasi Logika dengan Neuron
- Definisi Formal untuk Artificial Neurons
- Learning Rule dari Perceptron

B. Adaptive Linear Neurons (Adaline)

- Meminimumkan cost function dengan Gradient Descent
- Stochastic Gradient Descent

Artificial Neural Networks

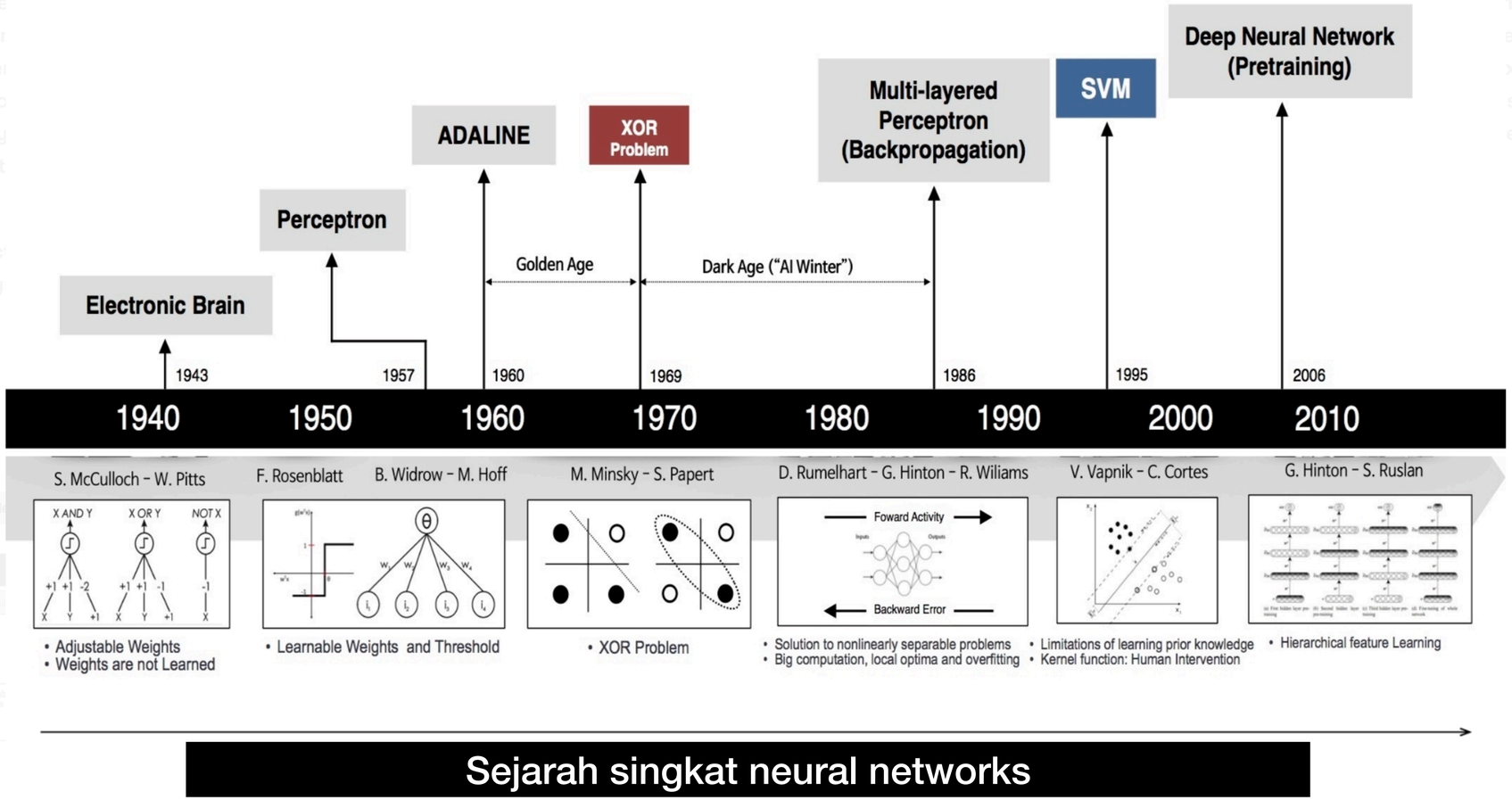
- Pada pertemuan sebelumnya dapat diambil kesimpulan dalam Machine Learning secara umum mempunyai langkah:
  1. Melihat pola
  2. Memodelkan pola dengan persamaan matematis
  3. Mencari parameter model dengan objective function atau cost function yang diminimumkan (proses training/learning)
  4. Implementasi menggunakan model dengan parameter optimum yang telah diketahui melalui proses training/learning

Bagian ini akan menjelaskan langkah 2 dan 3 untuk perceptron dan Adaline

Artificial Neural Networks

A. Artificial Neurons

- Artificial neurons diperkenalkan tahun 1943 oleh Warren McCulloch dan Walter Pitts,
- Tahun 1957 Rosenblatt menemukan perceptron
- Tahun 70an, NN tidak dapat memenuhi ekspektasi dapat menghasilkan mesin cerdas dan mengalami kemandekan panjang (AI Winter)
- Tahun 90-an, teknik machine learning lain yang lebih powerful ditemukan seperti Support Vector Machine (SVM) dll.
- Tahun 2010 Geoffrey Hinton dan mahasiswa PhD-nya memperkenalkan Deep Neural Network



Sejarah singkat neural networks

Sumber: dapat dilihat di link

Artificial Neural Networks

A. Artificial Neurons

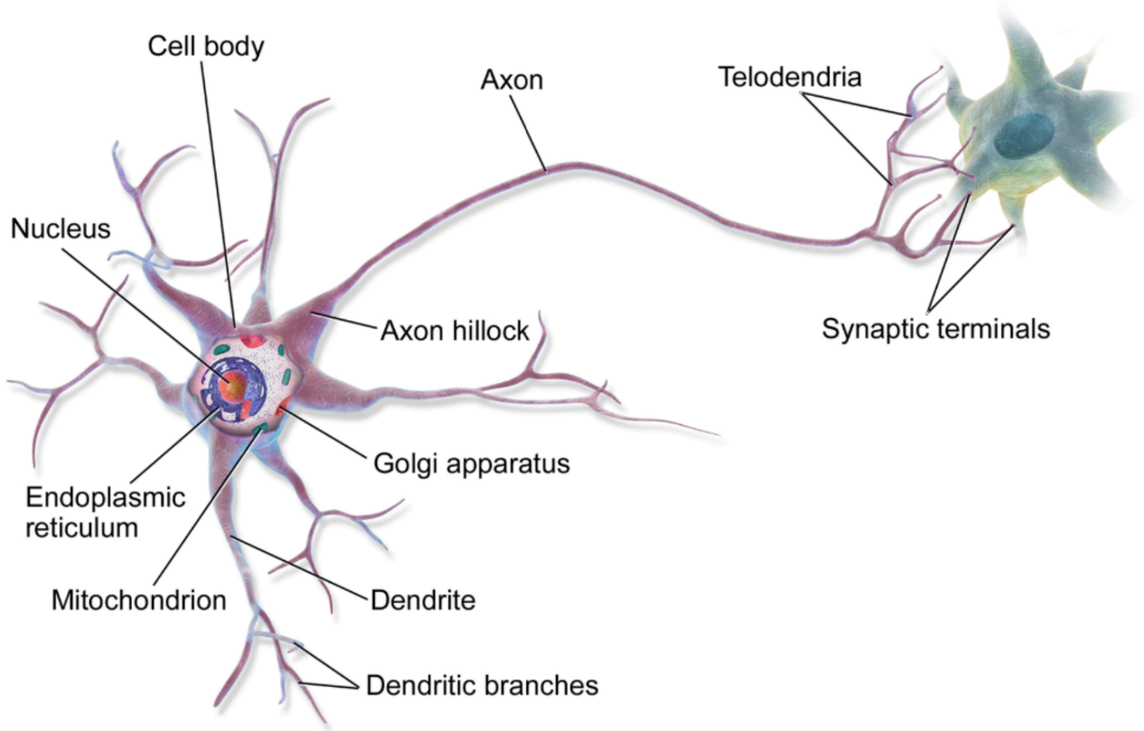
- Mulai 2010 terjadi gelombang ketertarikan lagi pada ANN, yang mungkin berbeda dengan kondisi di masa lalu, karena
  - Data melimpah untuk training ANN (untuk kasus data kompleks ANN mempunyai kinerja lebih baik dibanding teknik ML lain)
  - Peningkatan kemampuan komputasi sejak tahun 90-an, contoh produksi ***Graphical Processing Units*** (GPU) yang kemudian ditunjang dengan platform clouds yang dapat diakses oleh semua orang.
  - Algoritma training telah banyak mengalami perbaikan
  - Keterbatasan teoritis ANN ternyata bersifat ringan pada aspek praktis
  - ANN telah memasuki era perkembangan dan pendanaan yang terus menerus

Artificial Neural Networks

A.1. Neuron Biologis

- Inspirasi neural networks adalah neuron biologis, dengan representasi sel seperti pada gambar:
  - **Cell body** terdiri dari **Nucleus** (inti sel), percabangan **dendrite**, dan cabang lebih panjang **axon**
  - Pada ujung, axon bercabang kembali disebut **telodendria**, dengan pucuk cabang-cabang **synaptic terminals (synapses)**
  - Synapses terkoneksi dengan dendrit atau cell body neuron-neuron lain

Neuron biologis

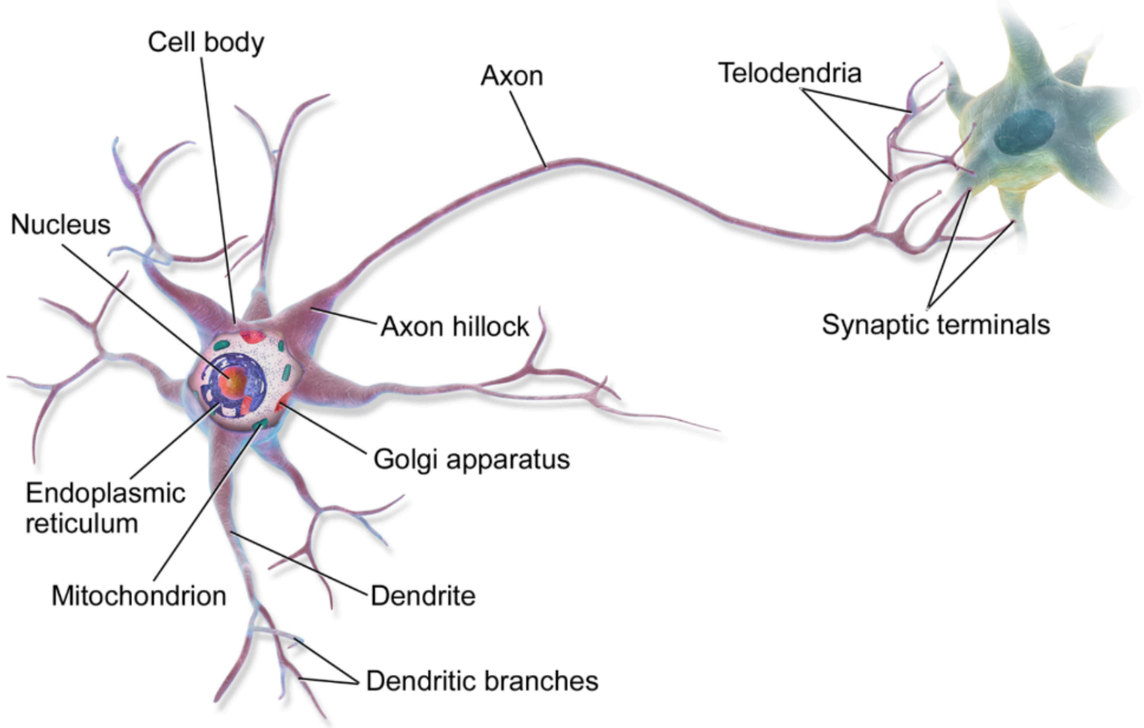


Artificial Neural Networks

A.1. Neuron Biologis

- Neuron-neuron biologis ini menghasilkan impuls-impuls listrik pendek yang disebut **action potentials (AP)**
- AP berjalan sepanjang axon dan membuat synapses memproduksi sinyal-sinyal kimia disebut dengan **neurotransmitters**
- Neuron-neuron penerima neurotransmitters akan melepaskan impuls-impuls listrik sendiri
- dst.

Neuron biologis



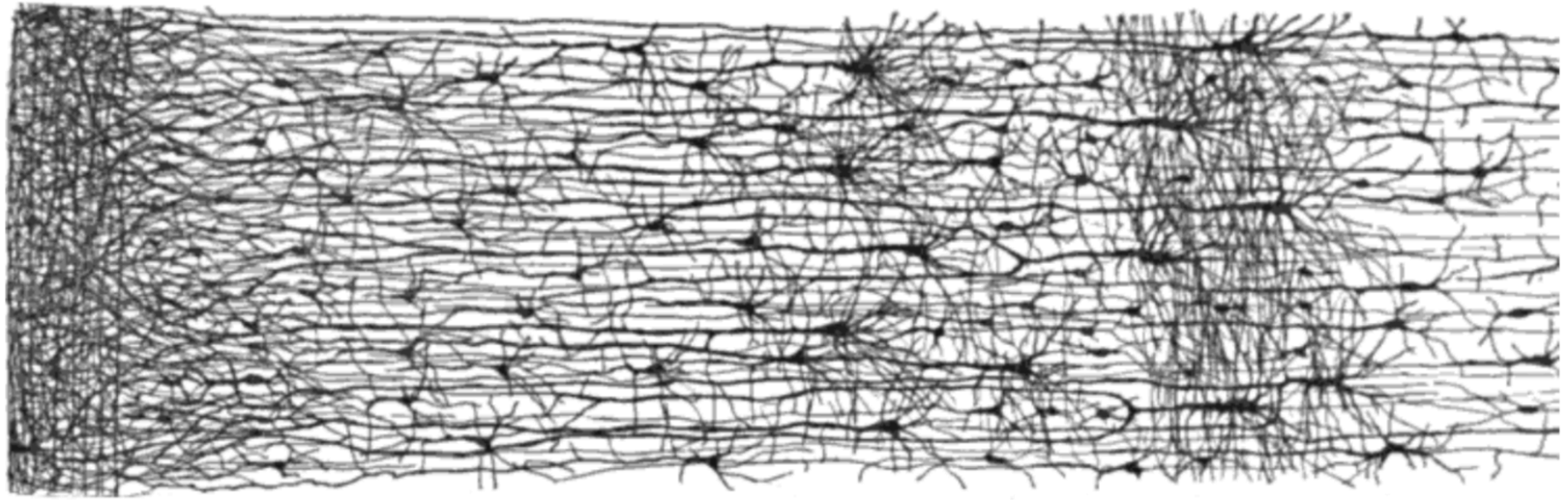
Artificial Neural Networks

A.1. Neuron Biologis

- Neuron-neuron individu yang sederhana diorganisir ke dalam jaringan yang sangat besar berjumlah miliaran.
- Masing-masing neuron biasanya terkoneksi dengan ribuan neuron lain
- Komputasi sangat kompleks dapat dilakukan dengan sebuah jaringan neuron-neuron sederhana

- Arsitektur **Biological Neural Networks** (BNN) masih merupakan subjek riset yang aktif.
- Dalam **cerebral cortex**, bagian terluar otak kita (lihat gambar), neuron-neuron tersusun pada layer-layer yang berurutan

Beberapa lapisan-lapisan pada BNN (cortex manusia)



Artificial Neural Networks

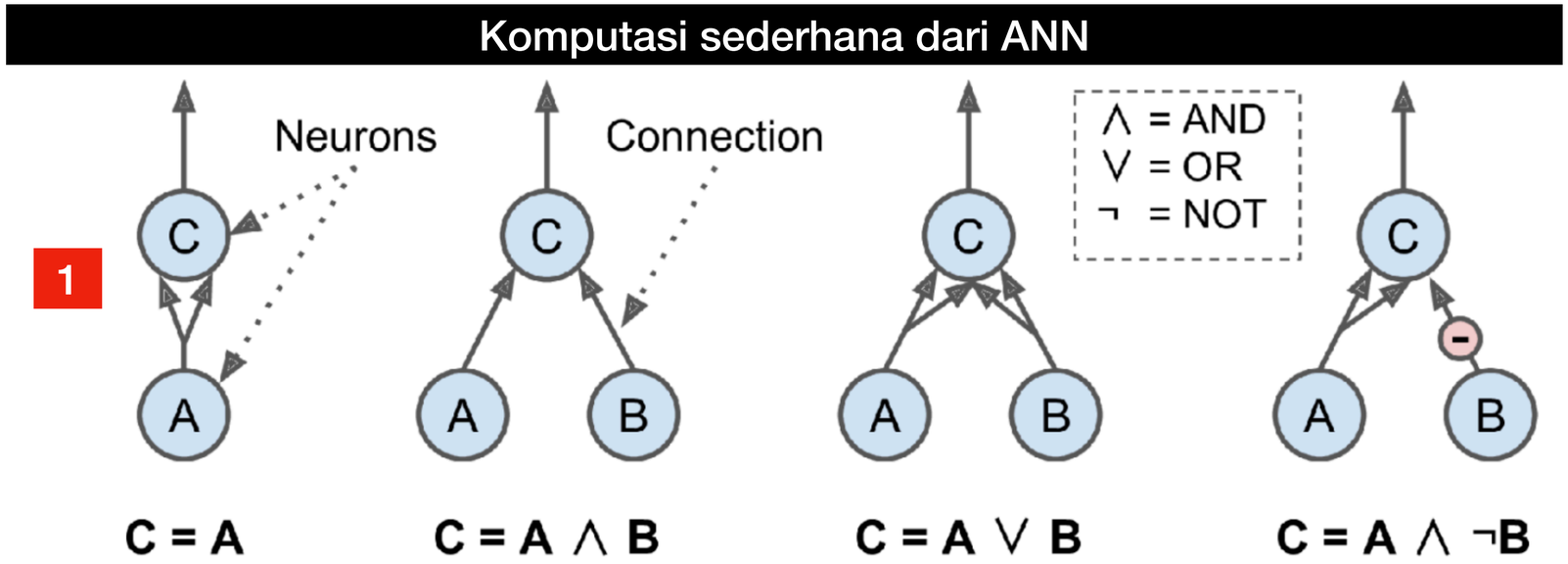
A.2. Komputasi Logika dengan Neurons

- McCulloch dan Pitss mengajukan model yang sangat sederhana dari neuron biologis, disebut **artificial neuron** (AN)
- AN mempunyai satu atau lebih input biner (on/off)
- AN mengaktivasi outputnya ketika jumlah input aktif lebih besar jumlah tertentu
- Dengan model sederhana tersebut, dapat dibuat jaringan AN yang menghitung operasi logika yang diinginkan

Artificial Neural Networks

A.2. Komputasi Logika dengan Neurons

- Contoh “sebuah neuron teraktivasi ketika sedikitnya dua input aktif”:

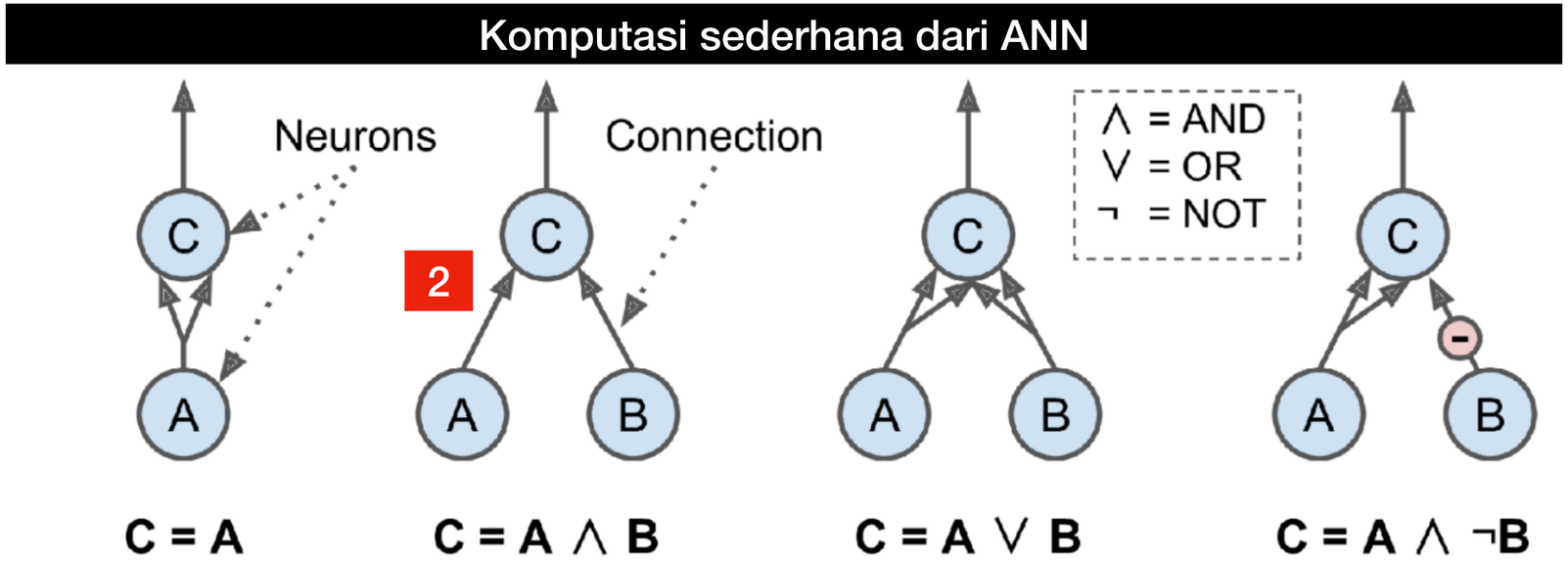


- Jaringan 1 adalah **fungsi identitas**:
  - Neuron A diaktivasi —> C teraktivasi
  - JIka neuron A off —> C off juga

Artificial Neural Networks

A.2. Komputasi Logika dengan Neurons

- Contoh “sebuah neuron teraktivasi ketika sedikitnya dua input aktif”:

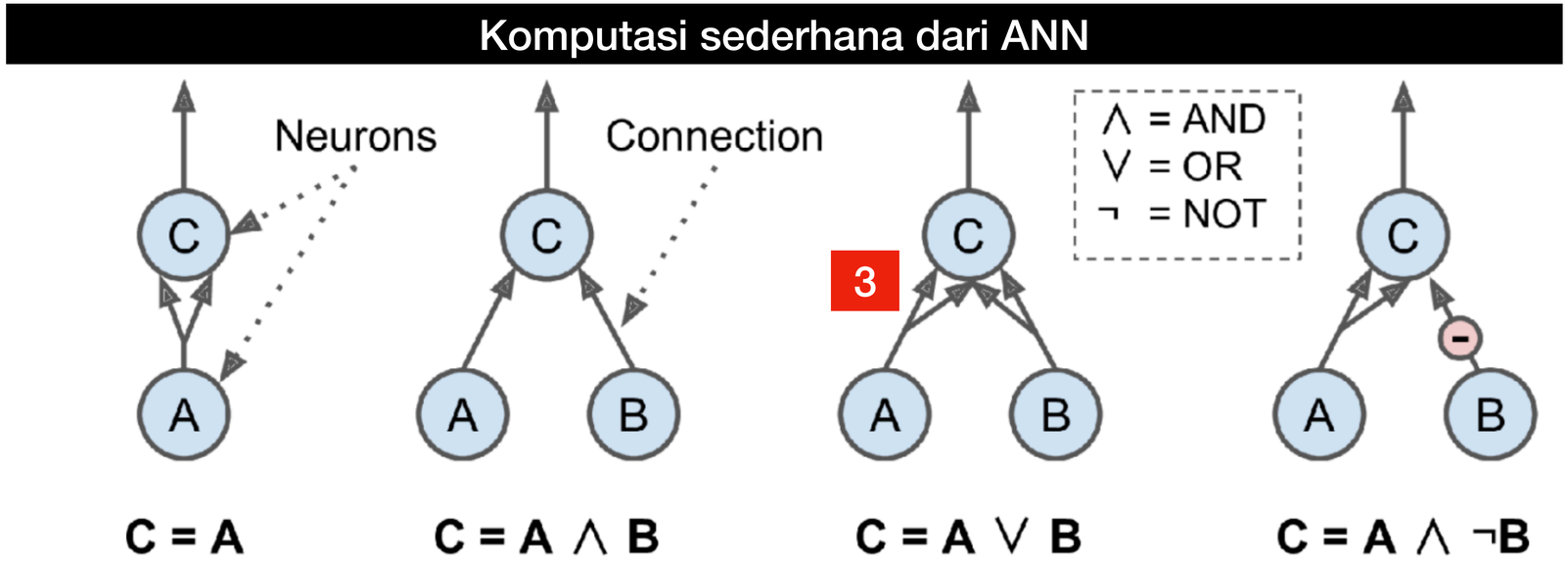


- Jaringan 2 adalah **logika AND**:
  - Jika A dan B aktif —> C teraktivasi
  - Jika hanya A atau B aktif, atau keduanya off —> C off

Artificial Neural Networks

A.2. Komputasi Logika dengan Neurons

- Contoh “sebuah neuron teraktivasi ketika sedikitnya dua input aktif”:

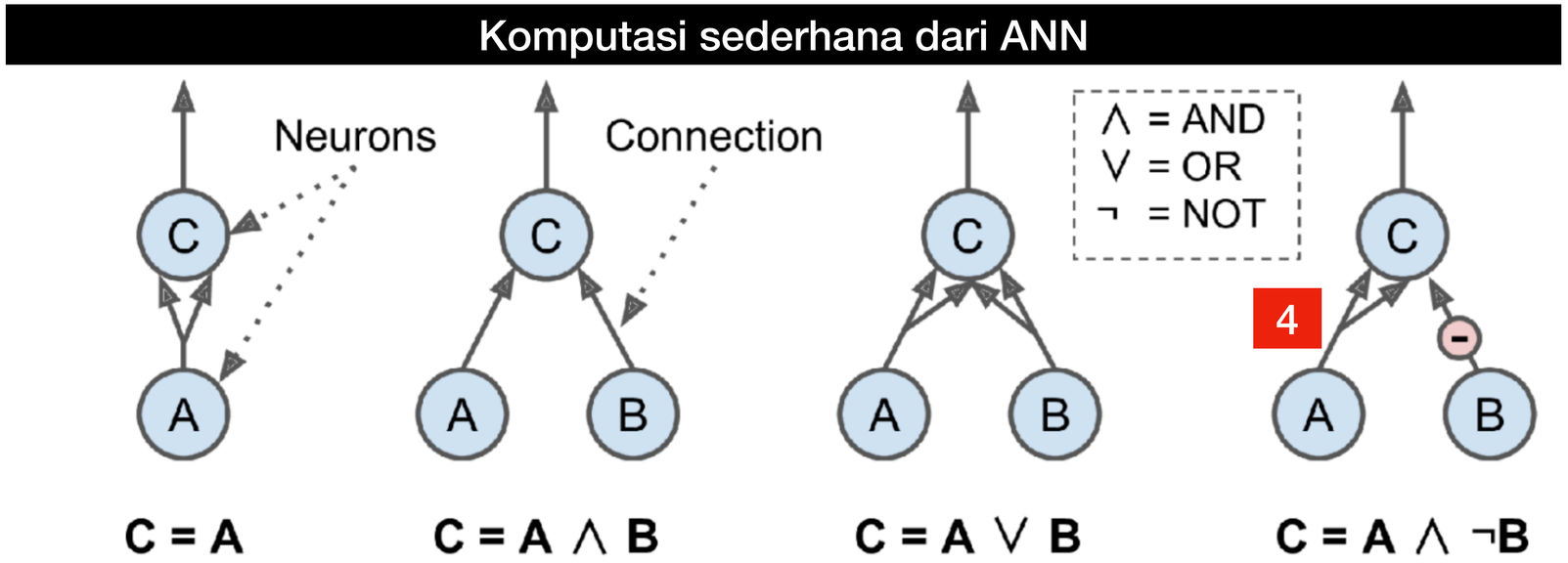


- Jaringan 3 adalah **logika OR**:
  - Jika A aktif, atau B aktif, atau keduanya aktif —> C teraktivasi
  - Jika keduanya off —> C off

Artificial Neural Networks

A.2. Komputasi Logika dengan Neurons

- Contoh “sebuah neuron teraktivasi ketika sedikitnya dua input aktif”:



- Jaringan 4 adalah **logika NOT** (catatan: ada kesalahan gambar, A ke C seharusnya satu input):
  - Jika A aktif dan B off —> C teraktivasi
  - Jika A aktif dan B Aktif —> C off

  Maka jika A terus aktif, C adalah NOT dari B

Artificial Neural Networks

A.3. Definisi Formal untuk Artificial Neuron

Bagaimana mendefinisikan secara formal (memodelkan) neuron atau perceptron?

Artificial Neural Networks

A.3. Definisi Formal untuk Artificial Neuron

- Ide artificial neuron secara formal dapat ditempatkan pada konteks persoalan klasifikasi biner (binary classification), dimana terdapat dua class, yaitu 1 (class positif) dan -1 (class negatif)

- Jika terdapat sebuah variabel z yang disebut **net input**, sebagai kombinasi linier dari **harga input** $\mathbf{x}$ dan pasangan **vektor pembobot (weights)** $\mathbf{w}$

$$z = w_1 x_1 + \ldots + w_m x_m \quad \text{atau} \quad z = \mathbf{w}^T \mathbf{x}$$

dimana

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Artificial Neural Networks

A.3. Definisi Formal untuk Artificial Neuron

- Didefinisikan sebuah fungsi keputusan $\phi(z)$:
  - dimana jika hasil net input z dari sampel ke-i, $\mathbf{x}^{(i)}$, lebih besar dari ambang batas θ, maka akan diprediksi sebagai class 1 $\left(\phi(z) = 1\right)$
  - atau jika sebaliknya maka akan diprediksi sebagai class -1 $\left(\phi(z) = -1\right)$

Pada algoritma **perceptron**, fungsi keputusan $\phi(z)$ adalah varian dari fungsi unit step (**unit step function**):

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta, \\ -1 & \text{otherwise.} \end{cases}$$

Artificial Neural Networks

A.3. Definisi Formal untuk Artificial Neuron

- Bentuk lebih sederhana diperoleh jika ambang batas (threshold) θ dibawa ke sebelah kiri persamaan dan mendefinisikan pembobot $w_0 = -\theta$ dan $x_0 = 1$ sehingga net input z

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \mathbf{w}^T \mathbf{x}$$

- Fungsi unit step dapat dituliskan kembali sebagai:

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Pada literatur ML, threshold negatif atau pembobot $w_0 = -\theta$ biasanya disebut dengan

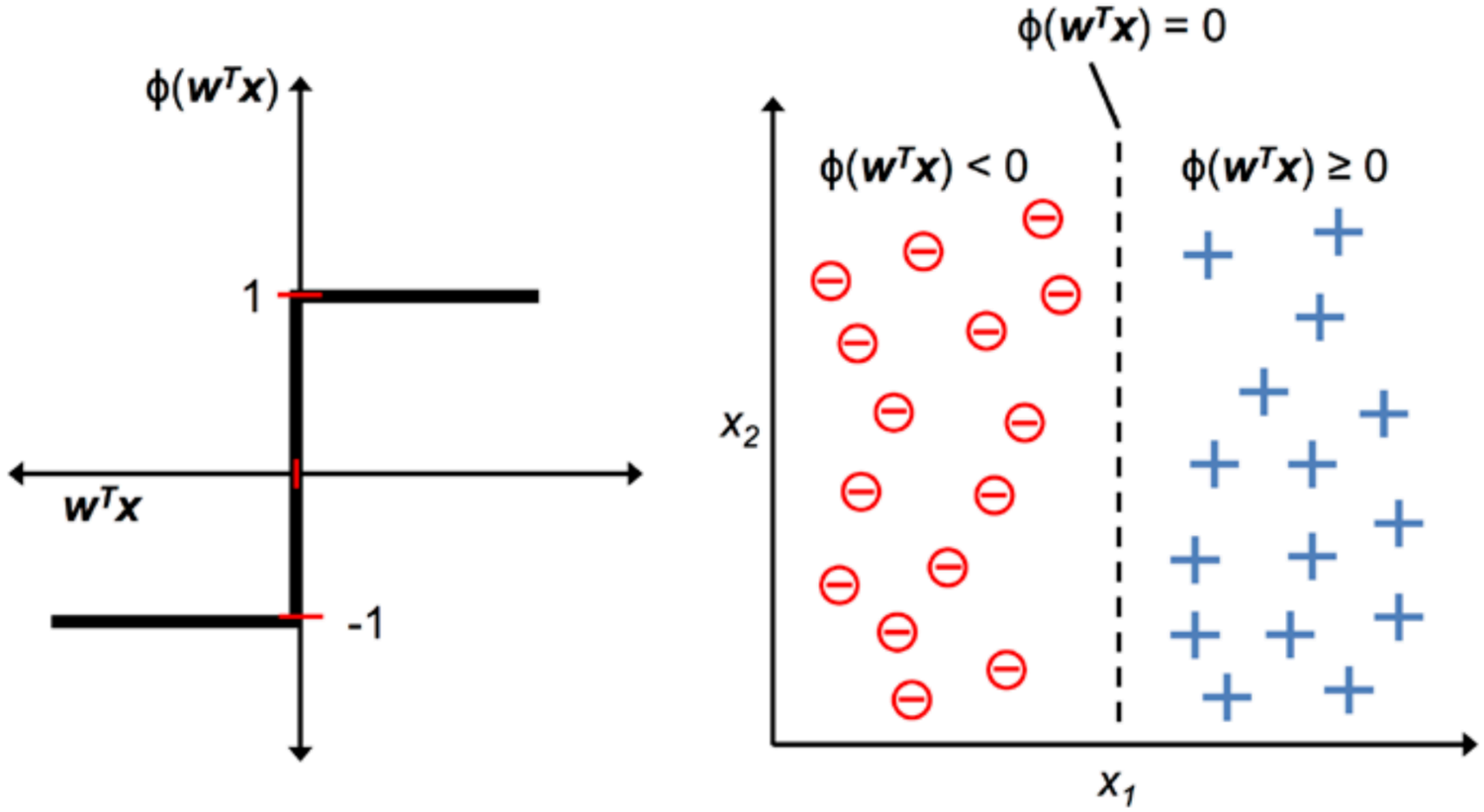
Bias Unit

Artificial Neural Networks

A.3. Definisi Formal untuk Artificial Neuron

- Gambar menunjukan net input z
  - diperas menjadi output biner (-1 atau 1) dengan menggunakan fungsi keputusan
    $\phi(z) = \phi(\mathbf{w}^T\mathbf{x})$
  - digunakan untuk mendiskriminasi antara dua class yang terpisah secara linier (**linearly separable classes**)

Net input menjadi output biner (kiri) dan fungsi keputusan perceptron (kanan)



Artificial Neural Networks

A.4. Learning Rule dari Perceptron

Bagaimana melakukan learning (training) dari perceptron?

Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Aturan learning (**learning rule**) perceptron awal dari Rosenblatt cukup sederhana:
  1. Inisialisasi bobot-bobot (weights) = 0 atau bilangan acak kecil
  2. Untuk tiap sampel training $\mathbf{x}^{(i)}$ maka
     (a) Hitung harga output $\hat{y}$ (label class yang diprediksi oleh fungsi unit step)
     (b) Update masing-masing bobot w_j pada vektor pembobot $\mathbf{w}$, dengan

$$w_j := w_j + \Delta w_j,$$

$$\text{dimana} \quad \Delta w_j = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

η : learning rate (konstan antara 0 s/d 1)
$y^{(i)}$: true class label (dari label sampel training ke-i
$\hat{y}^{(i)}$: predicted class label (class hasil prediksi)

Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Catatan: Semua bobot w_j pada vektor pembobot $\mathbf{w}$ diupdate secara bersamaan, dimana predicted class label $\hat{y}^{(i)}$ tidak dihitung kembali sebelum semua pembobot diupade melalui harga update yang bersesuaian.

- Contoh: untuk dataset dua dimensi, update:

$$\Delta w_0 = \eta \left(y^{(i)} - \hat{y}^{(i)} \right)$$

$$\Delta w_1 = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_1^{(i)}$$

$$\Delta w_2 = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_2^{(i)} .$$

Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Ilustrasi cara kerja learning rule perceptron

➡ Dua skenario ketika perceptron memprediksi label class dengan betul:

- $y^{(i)} = -1, \quad \hat{y}^{(i)} = -1, \quad \Delta w_j = \eta\left(-1-(-1)\right)x_j^{(i)} = 0$
- $y^{(i)} = 1, \quad \hat{y}^{(i)} = 1, \quad \Delta w_j = \eta\,(1-1)\,x_j^{(i)} = 0.$

Bobot-bobot tidak berubah karena update $w_j = 0$

➡ Dua kasus ketika perceptron salah memprediksi label class:

- $y^{(i)} = 1, \quad \hat{y}^{(i)} = -1, \quad \Delta w_j = \eta\left(1-(-1)\right)x_j^{(i)} = \eta(2)x_j^{(i)}$

  Bobot-bobot didorong menuju target class positive
- $y^{(i)} = -1, \quad \hat{y}^{(i)} = 1, \quad \Delta w_j = \eta\,(-1-1)\,x_j^{(i)} = \eta(-2)x_j^{(i)}$

  Bobot-bobot didorong menuju target class negative

Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Untuk memahami peran faktor pengali $x_j^{(i)}$, kita asumsikan

  $\hat{y}^{(i)} = -1, \quad y^{(i)} = +1, \quad \eta = 1$, dan $x_j^{(i)} = 0.5$,

  - Maka, penambahan bobot $\Delta w_j = (1-(-1))0.5 = 1$, dan net input $x_j^{(i)} \times w_j$ akan lebih positif pada saat selanjutnya sampel ini ditemui
  - sehingga kemungkinan selanjutnya mengklasifikasikan menjadi $+1$ (keputusan benar) semakin besar

Jika pada kondisi yang sama tetapi $x_j^{(i)}$ dinaikan menjadi $x_j^{(i)} = 2$, maka $\Delta w_j = (1-(-1))2 = 4$, net input akan semakin positif dan kemungkinan membuat keputusan benar $(+1)$ semakin besar pula

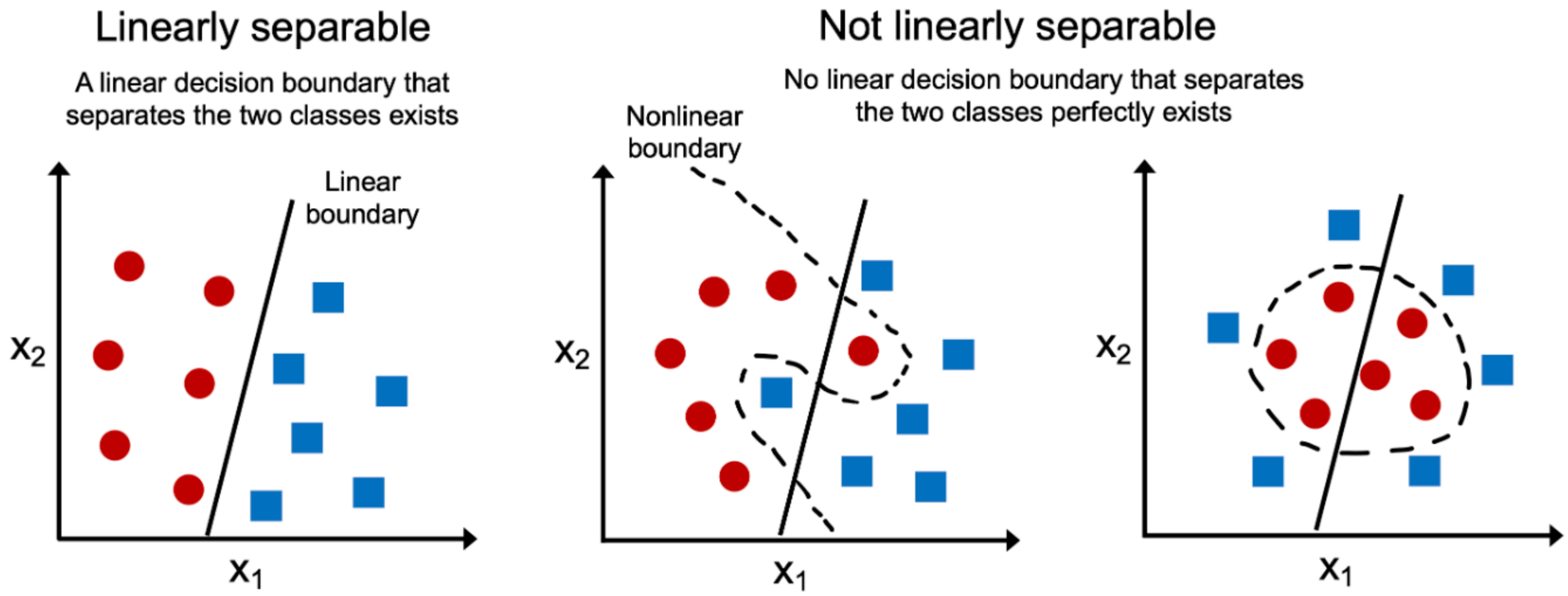
Update pembobot Δw_j adalah proportional dengan harga sampel training $x_j^{(i)}$

Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Konvergensi perceptron hanya dijamin jika kedua class adalah linearly separable dan learning rate cukup kecil

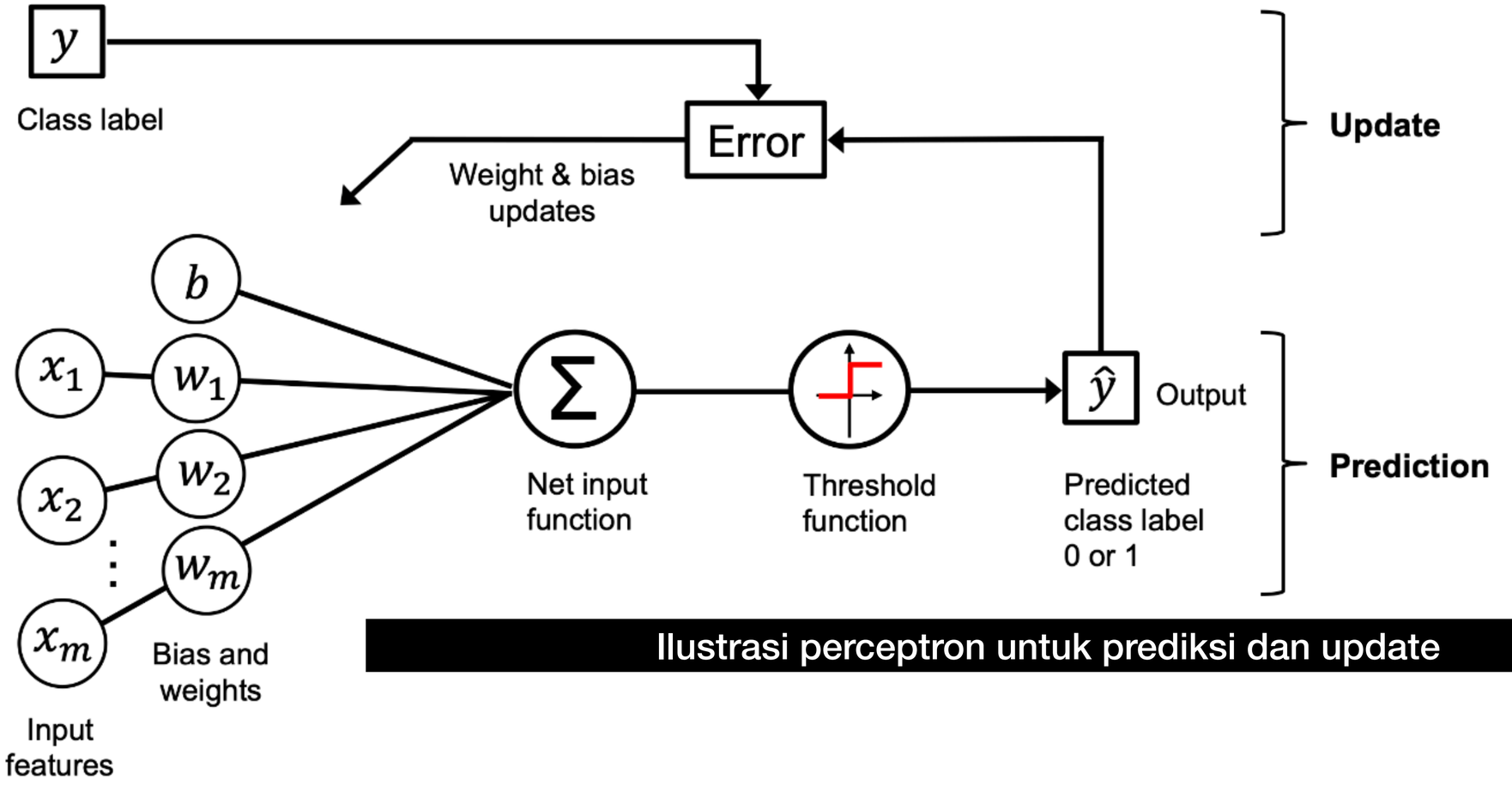
Ilustrasi data dua dimensi (x_1, x_2) yang linearly separable dan yang tidak



Artificial Neural Networks

A.4. Learning Rule dari Perceptron

- Diagram sederhana berikut menyimpulkan apa yang sudah dipelajari



Ilustrasi perceptron untuk prediksi dan update

Artificial Neural Networks

B. Adaptive Linear Neurons (Adaline)

- **ADAptive LInear NEurone (Adaline)** adalah tipe lain dari neural network satu layer
- Dipublikasikan oleh Bernard Widrow dan Tedd Hoff, beberapa tahun setelah algoritman perceptron dari Rosenblatt
- Adaline cukup menarik karena mengilustrasikan konsep mendefinisikan dan meminimalkan ***cost function*** kontinyu, yang merupakan dasar-dasar untuk memahami algoritma-algoritma lanjut baik klasifikasi maupun regresi

Artificial Neural Networks

B. Adaptive Linear Neurons (Adaline)

Bagaimana mendefinisikan secara formal (memodelkan) Adaline?

Artificial Neural Networks

B. Adaptive Linear Neurons (Adaline)

- Model serupa dengan Perceptron
- Perbedaan utama Adeline (Widrow-Hoff rule) dan perceptron Rosenblatt: Adaline mengupdate bobot berdasarkan fungsi aktivasi linier, sedangkan perceptron menggunakan fungsi step
- Fungsi aktivasi pada Adaline adalah fungsi identitas:

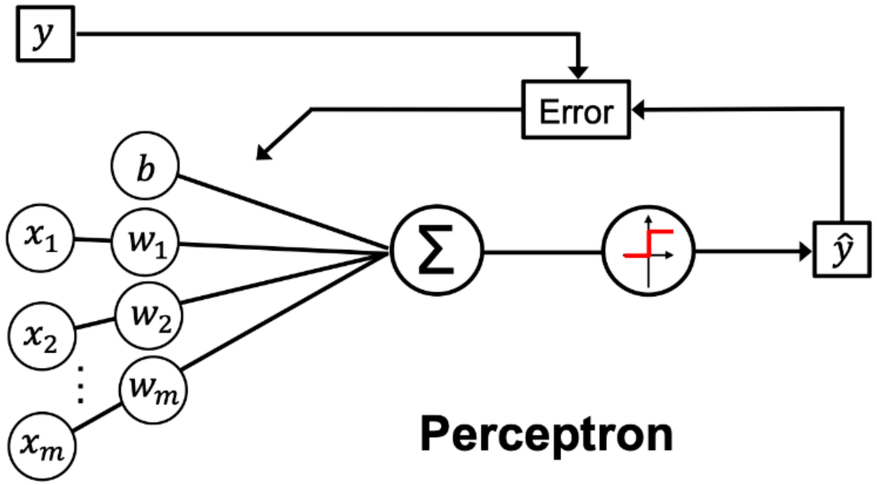
$$\phi\left(\mathbf{w}^T\mathbf{x}\right) = \mathbf{w}^T\mathbf{x}$$

Artificial Neural Networks

B. Adaptive Linear Neurons (Adaline)

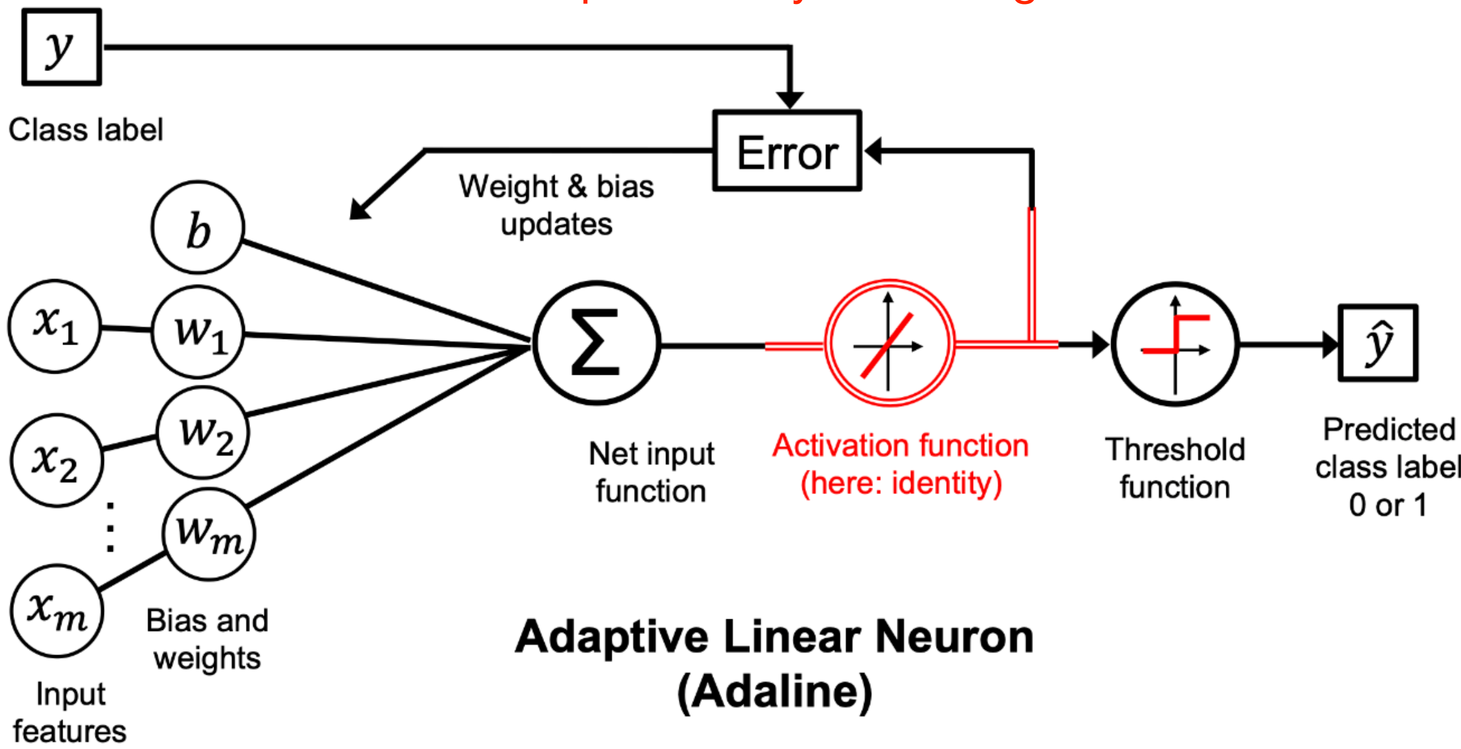
Perbedaan Antara perceptron dan Adaline terutama untuk fungsi aktivasi

Error: membandingkan *true class label* dan *predicted class label*



Perceptron

Error: membandingkan *true class label* dan output kontinyu dari fungsi aktivasi



Adaptive Linear Neuron (Adaline)

Artificial Neural Networks

B. Adaptive Linear Neurons (Adaline)

Bagaimana melakukan learning (training) dari Adaline?

Artificial Neural Networks

B.1. Meminimumkan *Cost Function* dengan *Gradient Descent*

- Salah satu unsur utama algoritma supervised ML adalah fungsi objektif (***objective function***) yang merupakan fungsi harga (***cost function***) yang harus diminimalkan
- Pada Adeline, cost function adalah **sum of squared errors** (SSE) antara luaran yang dihitung $\phi\left(z^{(i)}\right)$ dan true class label $y^{(i)}$:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i}\left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

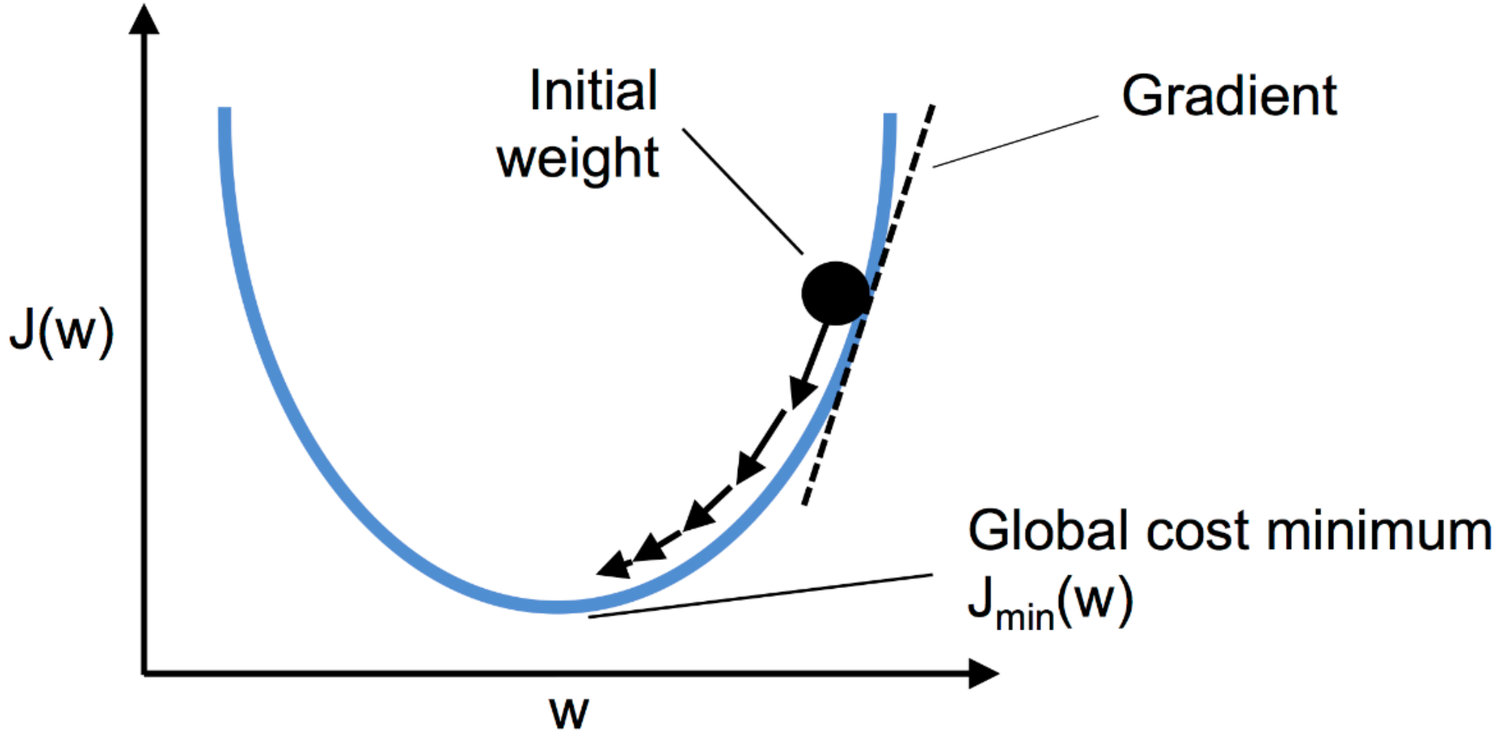
- Kelebihan Adaline:
  - Cost function dapat diturunkan karena merupakan fungsi aktivasi linier kontinyu (perceptron tidak)
  - Cost function bersifat convex, sehingga algoritma optimasi gradient descent bisa digunakan (perceptron tidak)

Artificial Neural Networks

B.1. Meminimumkan *Cost Function* dengan *Gradient Descent*

- Nilai-nilai pembobot pada vektor bobot $\mathbf{w}$ dicari sedemikian sehingga diperoleh cost $J(\mathbf{w})$ yang paling minimum
- Di setiap iterasi, diambil langkah yang berlawanan dengan arah gradient
- Ukuran langkah ditentukan oleh gradient dan learning rate

Ilustrasi gradient descent



Artificial Neural Networks

B.1. Meminimumkan *Cost Function* dengan *Gradient Descent*

- Ambil langkah berlawanan terhadap gradient $\nabla J(\mathbf{w})$:

$$\mathbf{w} := \mathbf{w} + \Delta w \quad \text{dimana} \quad \Delta w = -\eta \nabla J(\mathrm{w})$$

- Untuk menghitung gradient, *cost function* diturunkan terhadap masing-masing bobot (lihat module untuk detail penurunan parsial *cost function* SSE)

$$\frac{\partial J}{\partial w_j} = -\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

- Sehingga update tiap pembobot w_j pada vektor bobot $\mathbf{w}$ menjadi

$$w_j^{(next)} = w_j + \Delta w_j = w_j - \eta \frac{\partial J}{\partial w_j} = w_j - \eta \sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

Artificial Neural Networks

B.1. Meminimumkan *Cost Function* dengan *Gradient Descent*

- Catatan:
  - Meskipun learning rule Adaline serupa dengan perceptron, tapi harus diperhatikan bahwa $\phi\left(z^{(i)}\right)$ dengan $z^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)}$ adalah bilangan real dan bukan label class bernilai integer
  - Update pembobot dihitung berdasarkan semua sampel training set, maka disebut **Batch Gradient Descent** (BGD)

Artificial Neural Networks

B.2. *Stochastic Gradient Descent*

- Untuk dataset training yang sangat besar (orde jutant) yang biasanya terjadi dalam ML, maka BGD sangat tidak efektif karena harga komputasi akan besar
- BGD melakukan iterasi berkali-kali menggunakan keseluruhan dataset di setiap langkah menuju minimum global
- Alternatif dari BGD adalah **Stochastic Gradient Descent** (SGD), dimana update pembobot dilakukan satu persatu untuk setiap sampel training:

$$\Delta \mathbf{w} = \eta \left(y^{(i)} - \phi \left(z^{(i)} \right) \right) \mathbf{x}^{(i)}$$

Artificial Neural Networks

B.2. *Stochastic Gradient Descent*

- Kelebihan SGD dibanding BGD
  - SGD konvergen mencapai titik optimum lebih cepat dibandingkan BGD karena SGD hanya memakai satu sampel training setiap iterasi, sedangkan BGD keseluruhan dataset training
  - Error surface dari SGD mempunyai noise lebih besar dibanding BGD, sehingga SGD mempunyai kemungkinan lebih tinggi untuk keluar dari minimul lokal menuju minimum global

- Kompromi BGD dan SGD bisa dilakukan dengan implementasi BGD menggunakan subset data training lebih kecil (mini), mis. 10, 20, atau 32, dst. Metode ini disebut **Mini-Batch Gradient Descent** (MGD)